

LangGraph Pipeline

🕒 Created	@March 27, 2025 6:20 PM
🏷️ Tags	



Mangrove Intelligence Agent – System Manual

Overview

This system is a hybrid AI agent built with **LangGraph** and **LangChain**. It intelligently handles **two types of user questions**:

1. **Forecast** questions — real-time, location-specific mangrove health analysis
 2. **Research** questions — general, factual, or conceptual questions about mangroves
-

Agent Behavior Logic

When the user asks a question:

1. The agent first uses an LLM to **extract intent**:
 - Is it a *forecast* or *research* query?
 - Does the question mention a location?
 - Can we infer a U.S. state from that location?
 2. Then, depending on the detected **goal**, it either:
 - Activates a **LangGraph pipeline** (for forecast)
 - Routes to a **LangChain retrieval-based agent** (for research)
-

Forecast Agent (LangGraph)


➤ Goal

Deliver a 5–6 sentence summary of current and predicted mangrove conditions using:


- Real-time data (wind & water)
 - Satellite imagery (NDVI from Earth Engine)
 - Machine learning prediction (XGBoost)
-

Forecast Workflow Breakdown

1. `extract_intent_node`

- Parses `user_query`
- Returns: `goal` , `location` , `state`
-  **LangGraph Workflow: Routing** (decides `forecast` vs `research` path)

2. Router (Inside LangGraph)

- If `goal == forecast` → continue
- If `goal == research` → exit graph with flag
-  **LangGraph Workflow: Routing**

3. `select_stations`

- Loads NOAA station metadata from CSVs (per state)
- Uses LLM to rank best wind/water stations near given location

4. Parallel Data Fetching

These run in parallel after station selection:

- `resolve_gps` : Gets latitude/longitude from location
- `fetch_environmental_data` : Pulls real-time wind/water values from NOAA API
- `fetch_weekly_lags` : Gets 8 weeks of historical hourly data from NOAA, converts to weekly averages
- ⚡ **LangGraph Workflow: Parallelization**

5. `fetch_ndvi_lags`

- Uses Google Earth Engine to extract 8 weeks of NDVI data from MODIS imagery
- ⚡ **LangGraph Workflow: Parallelization** (with weekly lags above)


6. `build_feature_vector`

- Combines all time series into a unified DataFrame
- Adds lag features for wind, tide, and NDVI

7. `predict_ndvi`

- Applies scaler + XGBoost model to forecast NDVI for current week

8. `generate_summary`

- Uses LLM to summarize:
 - Wind speed
 - Water level
 - NDVI prediction
 - Original user intent
-  **LangGraph Workflow: Prompt Chaining** (LLM + context summary)

External Services Used

Node	Source / API
<code>resolve_gps</code>	OpenStreetMap Nominatim
<code>fetch_environmental_data</code>	NOAA Tides & Currents API
<code>fetch_weekly_lags</code>	NOAA Historical API (chunked)
<code>fetch_ndvi_lags</code>	Google Earth Engine (MODIS)
<code>build_feature_vector</code>	Pandas (time series wrangling)
<code>predict_ndvi</code>	XGBoost, Scikit-learn Scaler
<code>generate_summary</code>	LangChain Prompt Template + LLM

Research Agent (LangChain)

➤ Goal

Answer open-ended, non-location-specific questions about mangroves

Uses:

- LangChain's `ConversationalRetrievalChain`
- A Pinecone vector index of documents

- OpenAI Embeddings
- Memory to keep track of chat history

Example queries:

- "Why are mangroves important?"
- "What are common threats to mangroves?"

Modular Roles




Component	Purpose
<code>extract_intent_node</code>	Centralized logic for goal inference
<code>run_agent()</code>	Top-level function that wraps LangGraph and LangChain agents
<code>forecast_chain()</code>	Invokes LangGraph pipeline for forecasts
<code>run_research_chain()</code>	Invokes LangChain retriever agent
<code>memory</code>	Stores user & assistant messages across turns

Example Usage

```
response = run_agent("How are mangroves doing in Key West?")
# → Returns a real-time forecast summary using LangGraph

response = run_agent("What are mangroves?")
# → Returns an educational explanation using Pinecone + RAG
```

Developer Notes

- All forecast logic is **encapsulated in LangGraph** and benefits from:
 -  **Routing**
 -  **Parallelization**
 -  **Prompt Chaining**
- Easy debugging via `StateGraph.get_graph().draw_mermaid()`

- LangChain is used **only** for conversational memory + document retrieval

Extensibility Ideas


Idea	How
Add human-in-the-loop review	Use LangGraph's <code>pause</code> / memory checkpoint
Add streaming responses	Use LangGraph streaming + token events
Add more data sources	Plug in extra nodes (salinity, temperature, etc.)
Allow follow-up clarification	Use LangChain memory thread

7. `continuous_evaluator` (Optional)


- Inputs:** NDVI prediction + Actual NDVI (from delayed source)
- Loop:**
 - Evaluate accuracy
 - If bad, log feedback → optional retraining
- LangGraph Pattern:** Evaluator-Optimizer

flowchart TD

%% ——— ENTRY POINT ———


A[User Query] → B[extract_intent
 Structured LLM]


%% ——— ROUTING ———

B → |forecast| C[select_stations
 NOAA CSVs]

B → |research| Z[run_research_chain]

%% ——— PARALLEL FETCH AFTER STATION SELECTION ———

C → C1[resolve_gps
 OpenStreetMap]

C → C2[fetch_environmental_data
 NOAA real-time]

C → C3[fetch_weekly_lags
 NOAA history]

%% ——— NDVI PARALLEL BRANCH ———

C1 → D1[fetch_ndvi_lags
 Google Earth Engine]

%% ——— FAN-IN FOR FEATURE VECTOR ———

C3 → E[build_feature_vector
 Pandas]

D1 → E

%% ————— PREDICTION AND SUMMARY —————
E → F[predict_ndvi
🤖 XGBoost]
F → G[generate_summary
📝 LLM Summary]
G → H[FINAL OUTPUT]
%% ————— RESEARCH SIDE PATH —————
Z → ZZ[ConversationalRetrievalChain
📖 LangChain + Pinecone]

Details

Model Flow and Implementation Summary

Extract goal and location from user query

- ✅ `extract_intent_node`
- Parses `goal`, `location`, `state`
- Stores into state for downstream use

Goal Branch: summary or qa

- ✅ Supports `"summary"` → full forecast pipeline
- ❌ `"qa"` not yet implemented (planned in diagram)


Find nearby stations

- ✅ `select_station_by_location_node`
- Uses LLM to rank NOAA stations by proximity
- Outputs `station_ids` and `station_candidates`


Resolve GPS coordinates

- ✅ `resolve_gps_from_location_node`
- Geocodes `location, state` → outputs `gps` tuple


Fetch real-time wind and water data

-  `fetch_environmental_data_node`
- Pulls latest values from NOAA API
- Output: `environmental_data` dict


Fetch 8-week wind and water history

-  Performed inside `build_feature_vector_node`
- Uses `fetch_weekly_noaa_lags_chunked(...)`


Fetch 7-week NDVI history from MODIS

-  Also inside `build_feature_vector_node`
- Uses `get_cleaned_weekly_ndvi_series(...)`


Build feature vector from time series

-  `build_feature_vector_node`
- Constructs 23-dimensional input from current + lagged values
- Output: `feature_vector` + `feature_df` (for diagnostics)


Predict NDVI using XGBoost model

-  `predict_ndvi_node`
- Scales inputs using `scaler`, runs prediction
- Output: `ndvi_prediction` (float)

Generate natural language summary

-  `generate_summary_node`
- Interprets NDVI result and returns textual summary

Return final result

-  Final `State` includes:
 - User input

- Station metadata
- Real-time & historical data
- Feature vector
- Prediction
- Summary text

QA Goal Handling

- 🟡 Not yet implemented
- Can be added as a simple LLM QA branch